

# Physical computing vanuit Scratch met gebruik van scratchClient – **Beginners**

*Bestuur servo's, LEDs en meer vanuit Scratch  
met gebruik van RPi, Arduino, scratchClient*

Hans de Jong & Gerhard Hepp

Workshop voor de Pi And More en Maker Faire  
conferenties

# Deel 1: Introductie

# On-line video's

- Voor on-line video's, ga naar Youtube en zoek *scratchClient Tutorials*.
  - Video's zijn in het Engels.
  - Deze presentatie is in details bijgewerkt, maar video kan nog steeds gebruikt worden.
- ScratchClient heeft twee manieren om blokken te representeren:
  - Workshop mode: gebruik naam-waarde paren
  - Verbose mode: elke input en output heeft zijn eigen blok.
- In deze workshop gebruiken we Workshop mode
  - Zie appendix A hoe je de andere representatie kunt krijgen, die vaak handiger kan zijn.

# Organisatie van de workshop

- Welkom en introductie presentatie (5 min).
  - Daarna werkt iedereen voor zich.
  - Kies met je “werkplek partner” welke onderwerpen je wilt doen.
    - Er is meer materiaal dan wat behandeld kan worden in 110 minuten.
  - Talen: Nederlands, Engels en Duits
  - Aan het eind: kopieer het materiaal dat je gemaakt hebt naar een USB stick (als je het wilt bewaren)
  - Afbreken en opruimen (5 min)
- De voornaamste stappen:
    1. Krijg werkende hardware en scratchClient config met behulp van de scratchClient config tool.
    2. Plaats de onderdelen op het bord en test wat je gemaakt hebt.
    3. Schrijf wat code in Scratch
    4. Voeg meer hardware toe en werk de config file bij.
    5. Herhaal.

# Doelen

- Aan het eind van de workshop zou je in staat moeten zijn om een aantal van deze dingen te doen:
  - Thuis de opstelling reproduceren (aangenomen dat je de hardware hebt 😊 )
    - Hier kun je de files vinden: <https://github.com/hansdejonghv/scratchClient-Tutorials>
    - Of ga naar [www.github.com](https://www.github.com) en zoek naar *scratchClient*
  - Begrijp (afhankelijk van hoe ver je gekomen bent en hoe diep je erin gedoken bent)
    - Digitale output (b.v. een LED laten oplichten)
    - Digitale input (b.v. kijken naar een knop)
    - Analoge input (b.v. van een potentiometer)
    - Puls Breedte Modulatie (Pulse Width Modulation - PWM)
      - Om LEDs te dimmen
      - Om servo's te besturen
      - Om een zoemer te laten klinken
  - Begrijpen waar alle weerstanden voor zijn
  - In staat om scratchClient te configureren en te laten lopen
  - In Scratch een programma te maken om de fysieke input and output te besturen
  - De inputs and outputs in de gaten houden
- **Lol hebben!**

# Geen doel

- Het is geen doel om een compleet bruikbaar spel of ander programma te maken.
  - Je kunt dat thuis doen met je eigen creativiteit nu dat je weet hoe de verschillende componenten te besturen vanuit Scratch met gebruik van scratchClient.

# Voorbeeld van wat gemaakt kan worden met Scratch en scratchClient



- <https://www.youtube.com/watch?v=Qo1gnXNzhqE>

# Versies van Scratch

- scratchClient kan werken met
  - Scratch 1.4
  - De (voor Raspberry Pi) nieuwe Scratch 2
- Deze workshop is gemaakt voor Scratch 2
  - Scratch 2 op Raspberry Pi heeft een aantal fouten, maar daar zullen we omheen werken.
  - Als je wilt zien hoe het te doen voor Scratch 1.4, zie het eind van de presentatie in Appendix B.

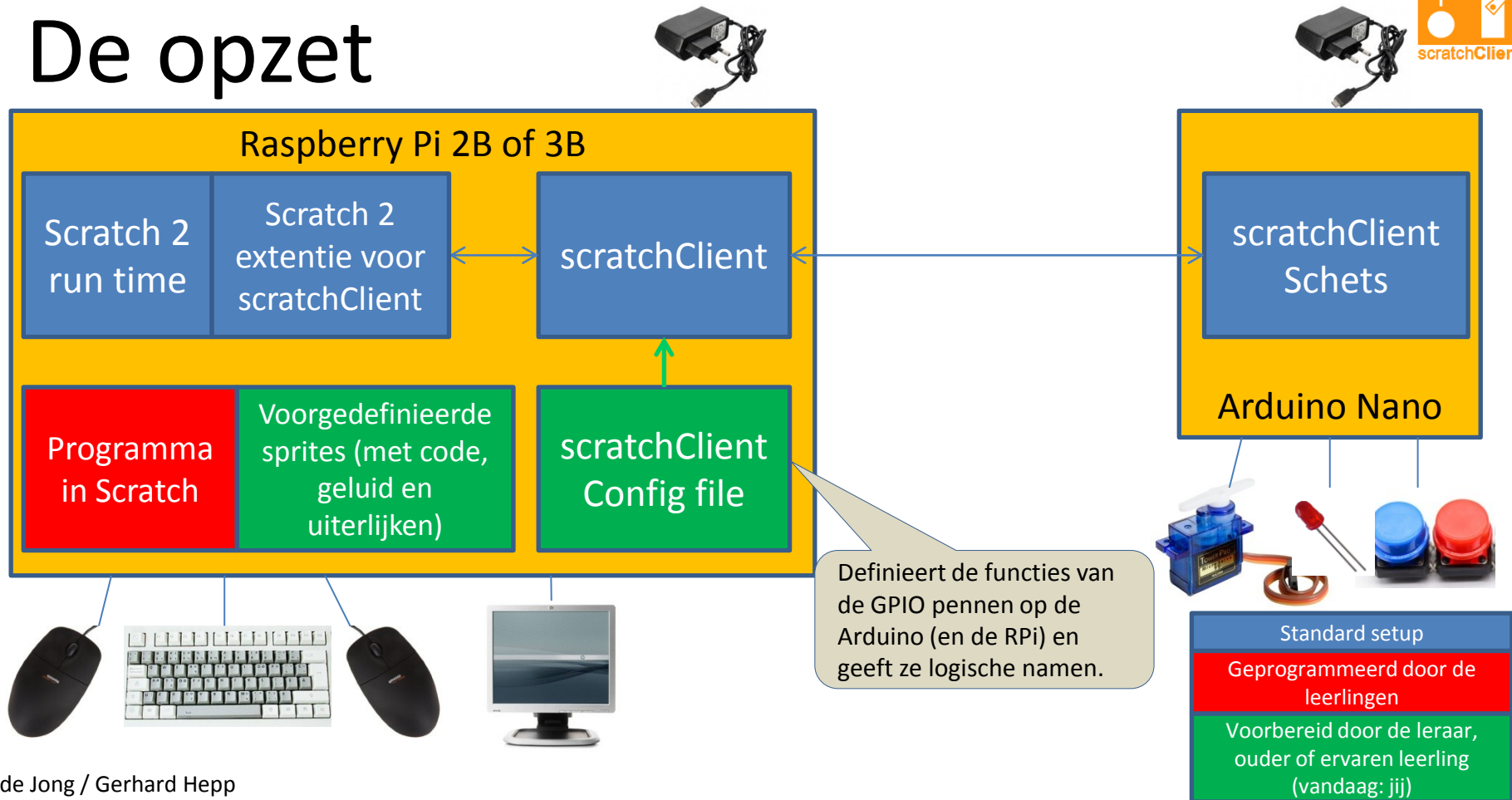


# Slechts een paar regels vandaag

- Plaats altijd weerstanden in serie met de componenten als dat is aangegeven.
  - Als je denkt dat dat niet nodig is, vertel het en we zullen de noodzaak uitlegge
- Als je de bedrading verandert
  1. Trek de USB kabel uit de Arduino Nano 
  2. Trek de 9 volt kabel eruit 
  3. Controleer, controleer tweemaal en controleer nogmaals dat de bedrading correct is.  
Je kunt componenten opblazen als je bedradingsfouten maakt!
  4. Zorg dat jullie **beiden** (4 ogen principe) overtuigd zijn dat de bedrading in orde is voordat je de kabels opnieuw aansluit.
  5. Na het wijzigen van een config file: start scratchClient opnieuw
- Als iets kapot gaat: we hebben extra materiaal
  - Stop s.v.p. **nooit** iets dat kapot is terug in de doos.



# De opzet



# Vaak gevraagd: waarom een Arduino gebruiken?



- Vraag: waarom niet gewoon de GPIO pennen van de Raspberry Pi gebruiken?
- Antwoord:
  - Arduino heeft analoog in en heeft hardware pulsbreedte modulatie (Pulse Width Modulation – PWM) voor een meer stabiele servobesturing.
  - Arduino gebruikt 5 volt. Raspberry Pi GPIO pennen zijn niet 5 volt tolerant.
  - Als kinderen fouten maken met de elektronica dan zullen ze eventueel een Arduino nano kloon van 2 euro opblazen i.p.v. een Raspberry Pi van 35 euro.
  - Maakt de opstelling transportabel, nu hoeft slechts een enkele USB plug te worden aangesloten en geen grote kabel op de GPIO pennen.

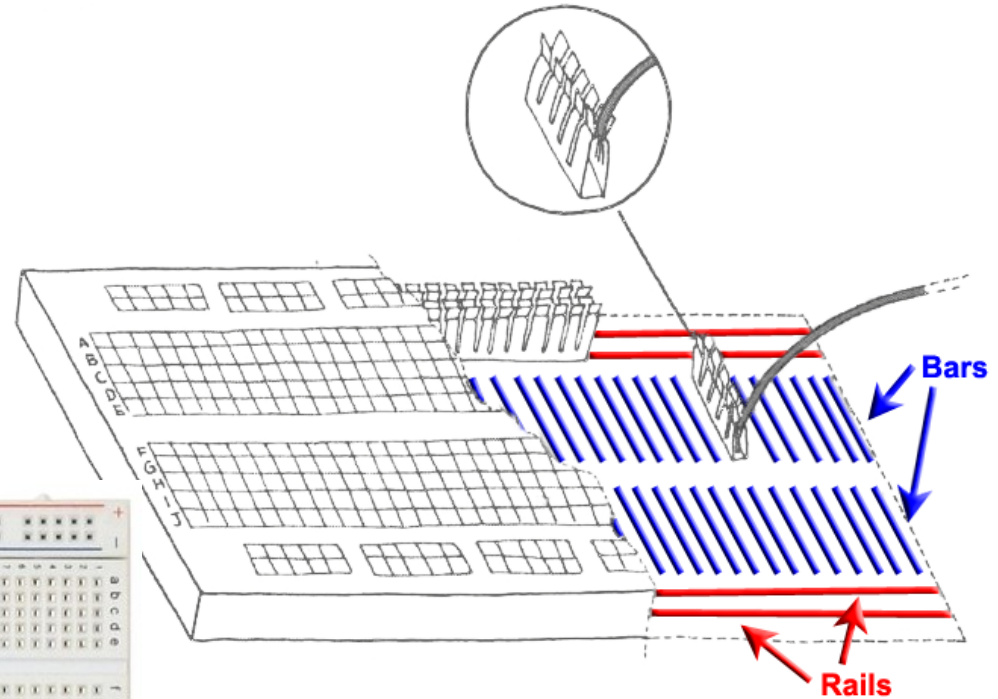
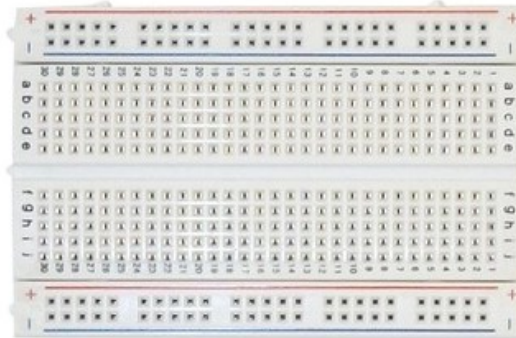
# Kies wat je wilt doen

- De workshop is voor iedereen, van beginner tot expert.
- Er is onvoldoende tijd om alles te doen, dus kies wat je gaat doen.
- Gele pagina's hebben achtergrond informatie en je kunt ze overslaan of er later naar kijken.
- Aanbeveling
  - Iedereen: leer hoe scratchClient te configureren met LEDs (digitaal uit) en knop (digitaal in)
  - Iedereen: probeer het uit in Scratch
  - Daarna: selecteer extra onderwerpen van afzonderlijke presentaties
    - Gevorderden
    - Geavanceerd
    - Expert

# Deel 2: Leer de componenten kennen

# Breadboard

- Wordt gebruikt om snel elektronische schakelingen te bouwen.
- Bekijk de twee rails voor + (VCC) and – (GND - aarde)
- Let op dat in elke kolom 2 blokjes met elk 5 met elkaar doorverbonden gaten zitten



# De Arduino Nano en het adapterbord

Analog poorten (de meeste ook  
bruikbaar als digitale poorten)

Reset knop

Voedingsaansluiting 9-12 volt

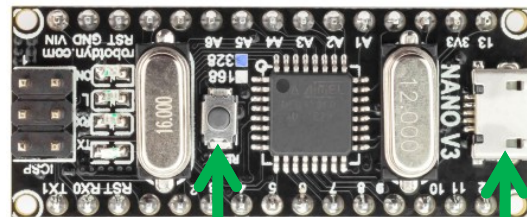
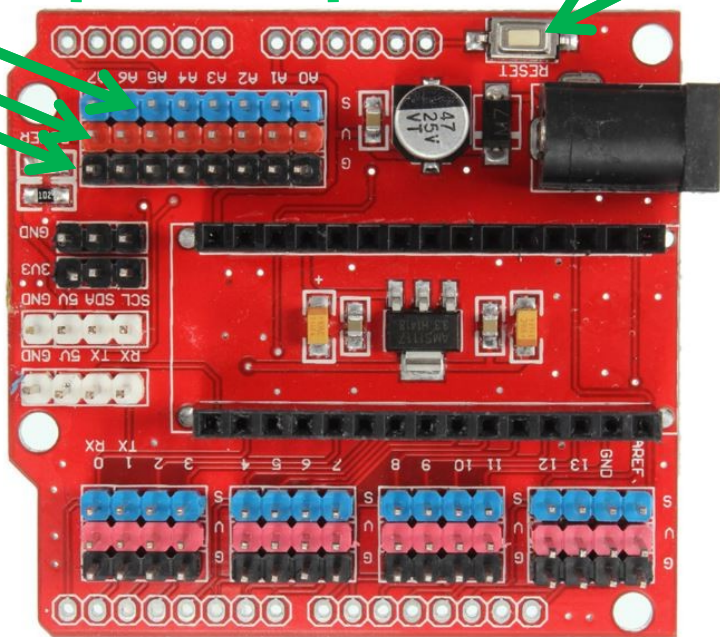
Arduino Nano met  
328P processor

Micro USB poort

Reset knop

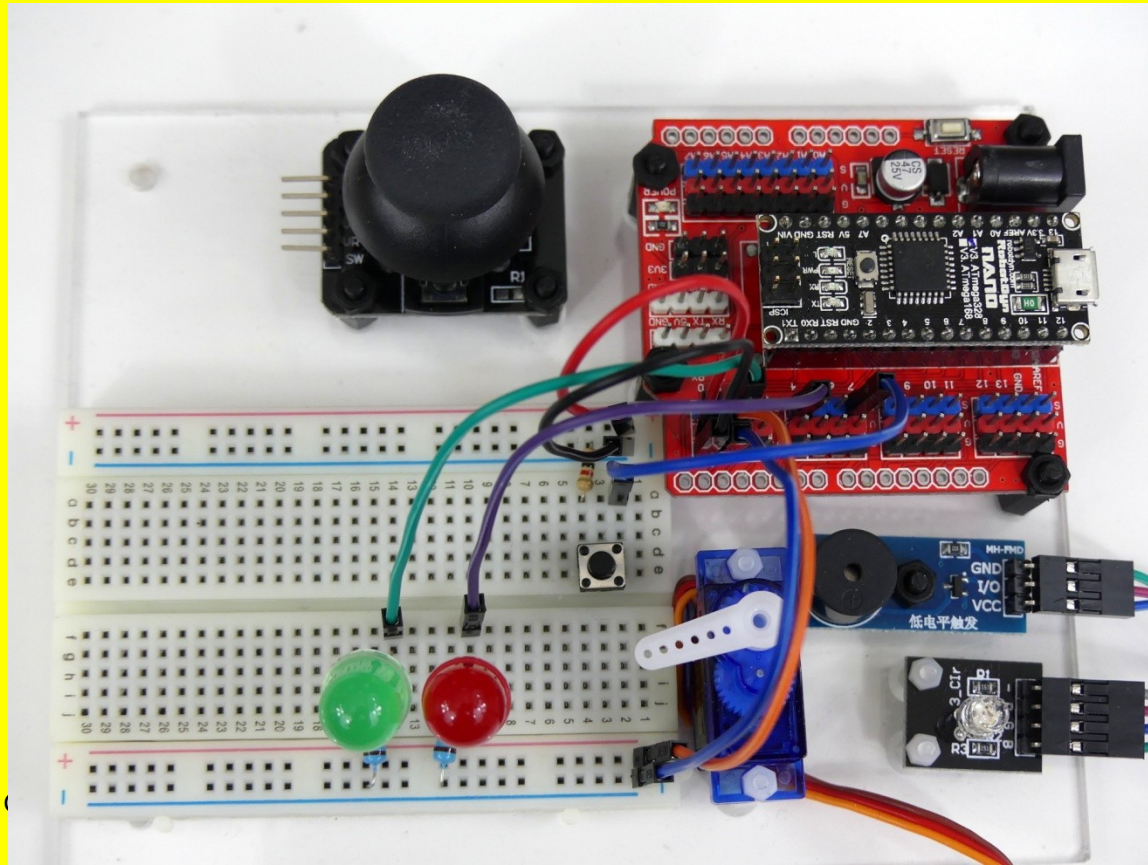
Digitale poorten

Per GPIO signaal 3 pennen:  
S (blauw = signaal)  
V (rood = VCC = +)  
G (zwart = GND = -)  
(heel handig om b.v.  
servo's aan te sluiten)





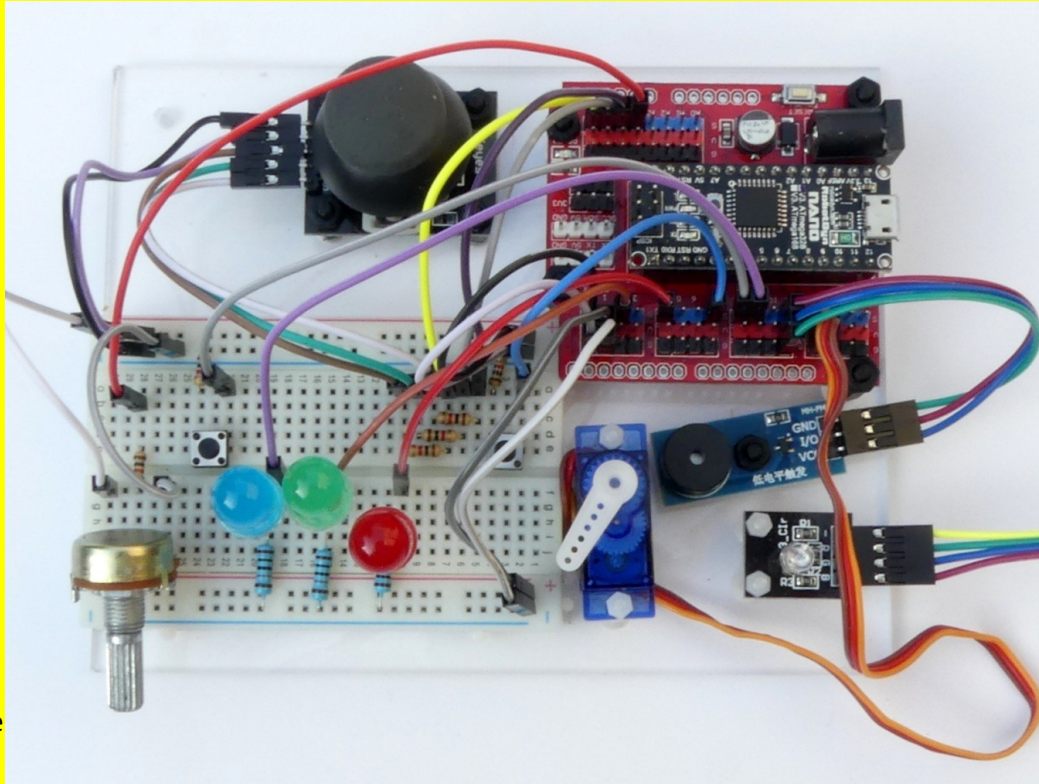
# Hoe het bord eruit ziet ...



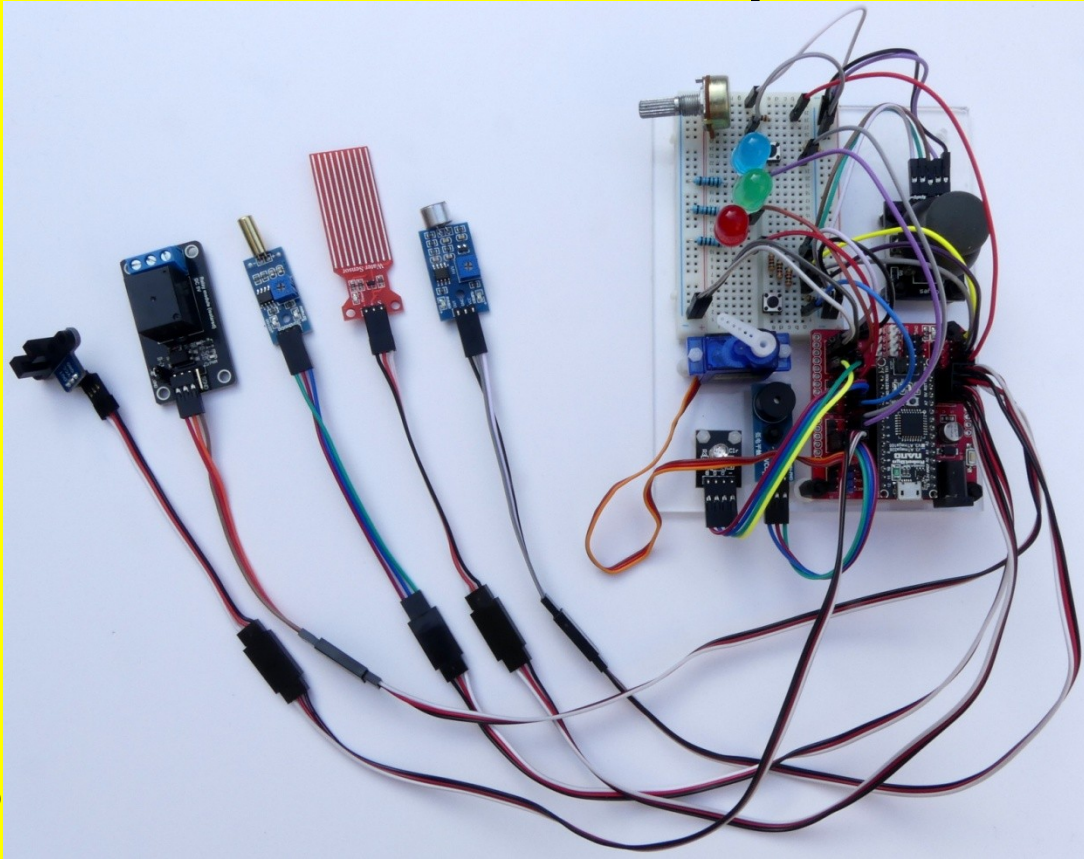
... aan het eind  
van de  
beginners  
workshop



# ... aan het eind van de gevorderden workshop ...



# ... en aan het eind van de geavanceerde workshop



# Deel 3: Het laden van de schets in the Arduino (in deze workshop: naar keuze, thuis: noodzakelijk)

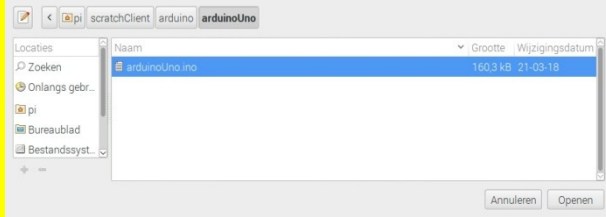
# Voorbereiden van het programmeren van de Arduino Nano




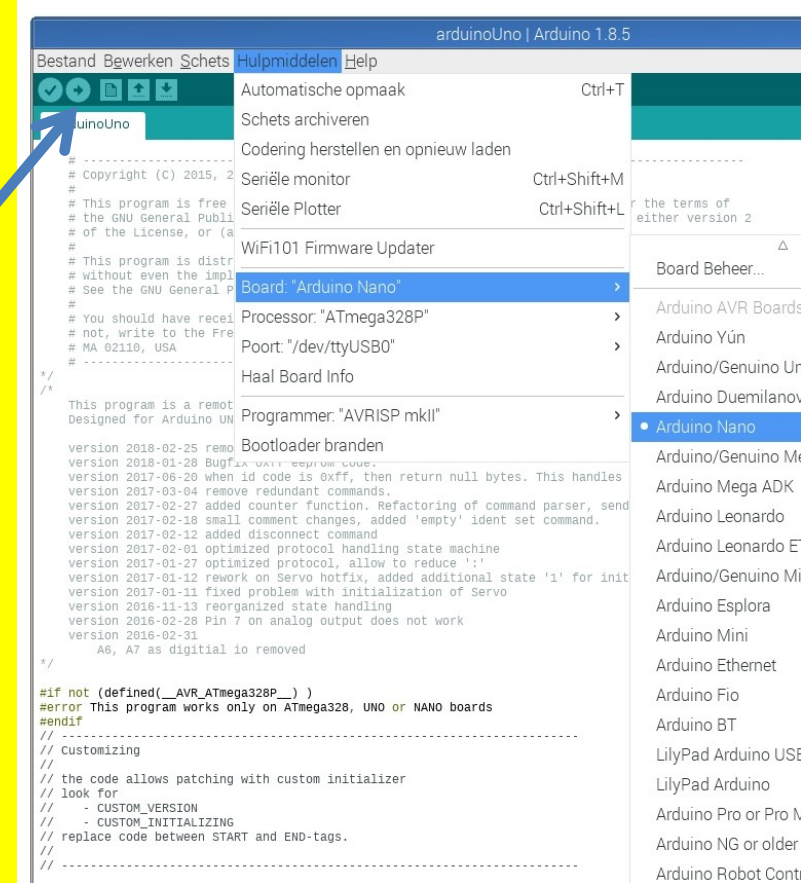
- De Arduino moet een programma draaien (in Arduino termen: schets) zo dat die kan communiceren met de Raspberry Pi en de berichten van scratchClient kan begrijpen.
- We moeten dat programma in de Arduino laden.
  - Echter, in de workshop is dat al gedaan, dus je kunt de volgende pagina overslaan, tenzij je het zelf wilt proberen.
  - Als je dit echter thuis doet, dan moet je het wel doen.

# ScratchClient laden in de Arduino

- Navigeer naar de scratchClient schets voor Arduino Uno in  
*/home/pi/scratchClient/arduino/arduinoUno*



- Dubbel klik erop om de Arduino IDE te openen
- Klik op *Tools* en zorg dat dit gezet is:
  - Board: Arduino Nano
  - Processor: Atmega328
  - Port: de poort waar de Arduino Nano is aangesloten (meestal /dev/ttyUSB0)
- Klik op de Upload button. 
- Wacht tot de Upload klaar is (zonder fouten).

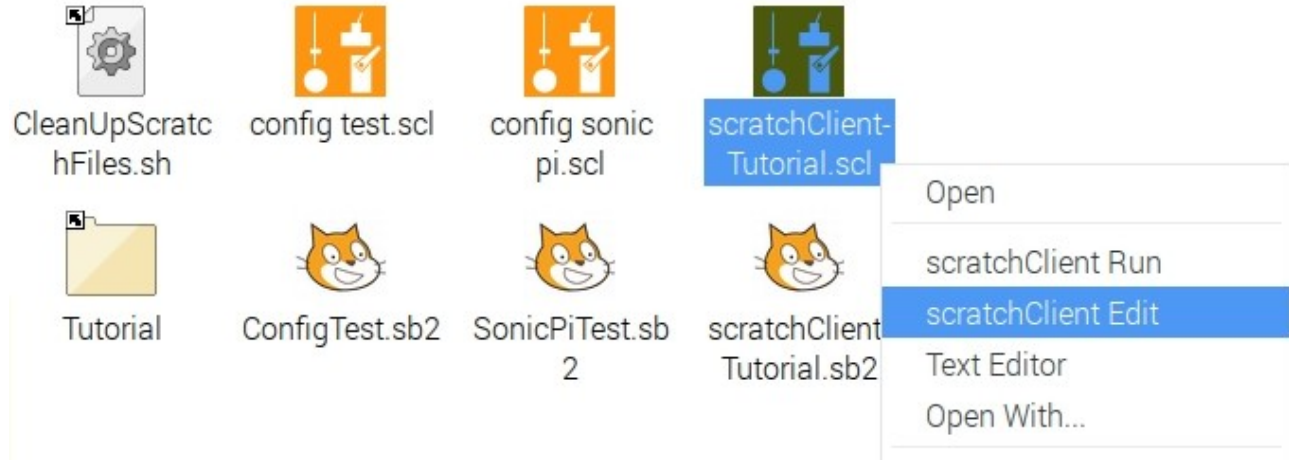


# Deel 4: Definiëren van de configuratie

Geef de pennen namen en definieer het doel van elke pen

# Start het config tool

- We hebben een lege config file op het bureaublad geplaatst:
  - *scratchClient-Tutorial.scl*
- Klik rechts op de file en kies *scratchClient Edit*





# Definieer de eerste config file

File Help

name: scratchCI Tutorial

| name | arduino | direction | function     | scratchName |
|------|---------|-----------|--------------|-------------|
| D0   |         | void      |              |             |
| D1   |         | void      |              |             |
| D2   |         | void      |              |             |
| D3   |         | void      |              |             |
| D4   |         | out       | output       | BigRedLED   |
| D5   |         | void      |              |             |
| D6   |         | void      |              |             |
| D7   |         | in        | input_pullup | Button      |
| D8   |         | void      |              |             |
| D9   |         | void      |              |             |
| D10  |         | void      |              |             |
| D11  |         | void      |              |             |
| D12  |         | void      |              |             |
| D13  |         | void      |              |             |
| A0   |         |           |              |             |
| A1   |         |           |              |             |
| A2   |         |           |              |             |
| A3   |         |           |              |             |
| A4   |         |           |              |             |
| A5   |         |           |              |             |
| A6   |         |           |              |             |
| A7   |         |           |              |             |

D4  
Direction: out  
Function: output  
scratchName: BigRedLED

D7  
Direction: in  
Function: input\_pullup  
scratchName: Button

Parameter serial.device: /dev/ttyUSB0

Parameter serial.device: /dev/ttyUSB0

Parameter ident.check: ☒

Parameter ident.pattern:

```

<!-- id = 'D7' direction = 'in' function = 'input_pullup' -->
<output_value name="outputD7">
  <sensor name="Button"/>
</output_value>

<extension>
  <io dir="out" id="D4"/>
  <io dir="in" id="D7" pullup="on"/>
</extension>

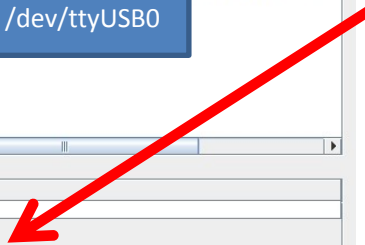
<parameter name="serial.device" value="/dev/ttyUSB0"/>
<parameter name="serial.baud" value="115200"/>

<!-- optional parameters for IDENT check -->
<parameter name="ident.check" value="yes"/>
<parameter name="ident.pattern" value=""/>
  
```

type id message

INFO empty ident.pattern connects only to arduino with empty ident

- **Dubbelklik** een cel om een drop down menu te krijgen
  - Eerst voor *direction* en dan voor *function*.
- Zorg ervoor alle pennen een naam te geven als je iets anders kiest dan *void* in *direction*.
  - Dus niet opslaan zolang je nog rode randen rond cellen ziet. scratchClient zal niet starten met zo'n config file.
- Het programma controleert verkeerde configuraties. Voorbeelden:
  - Pennen 0, 1 en 13 kunnen helemaal niet gebruikt worden
  - Analoog In kan alleen op A0 tot A7
  - Pennen A6 and A7 kunnen alleen voor Analoog In gebruikt worden
  - Pennen 9 en 10 kunnen niet voor PWM worden gebruikt als enige pen voor servo is geconfigureerd (zie later)



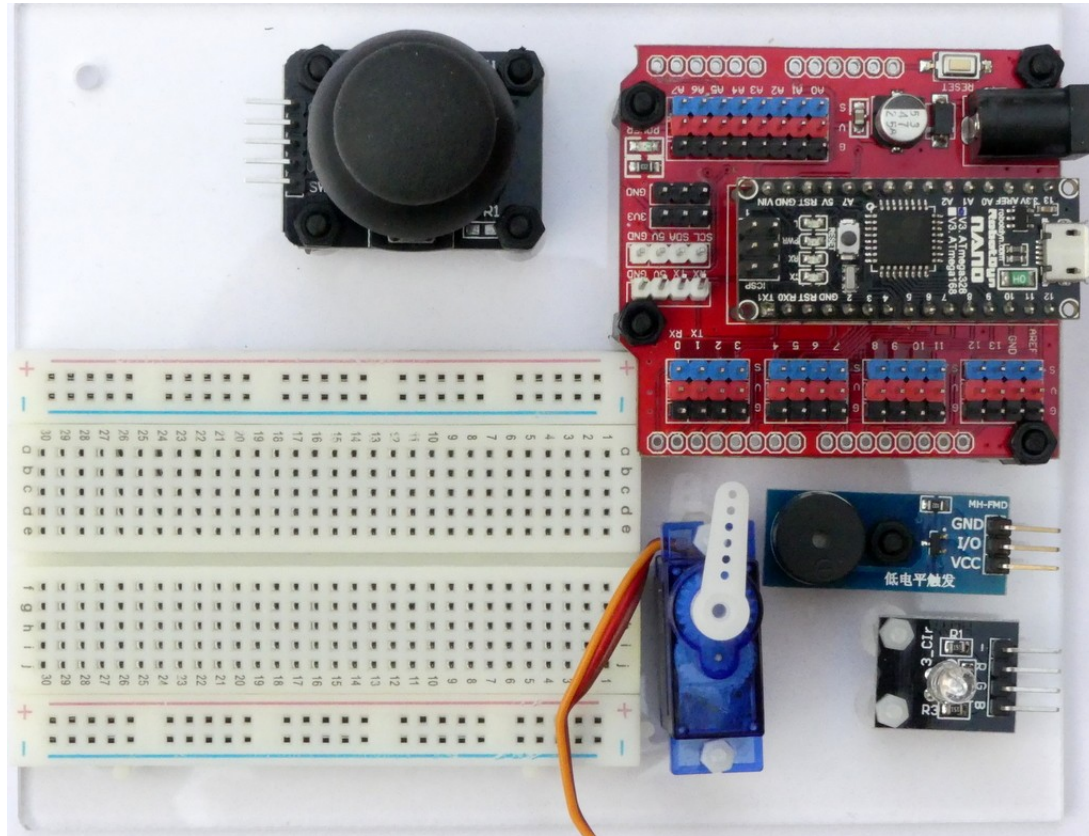


# Sla de config file op

- Sla de file op (ctrl-S of File → Save).
- **Waarschuwing:** sluiten van de file zonder op te slaan laat alle wijzigingen verloren gaan.
- **Laat het tool open** voor de volgende opgaven.

# Deel 5: Bedraad het bord en doe de eerste setup

# Leg het bord zo voor je

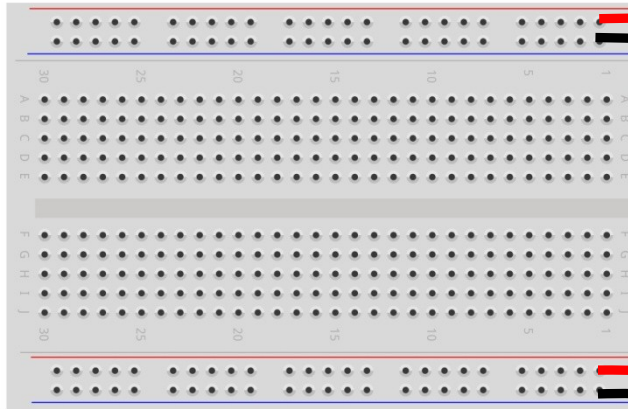
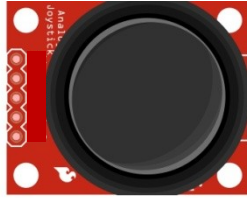


# Gebruik *korte* draden en gebruik de aangegeven gaten

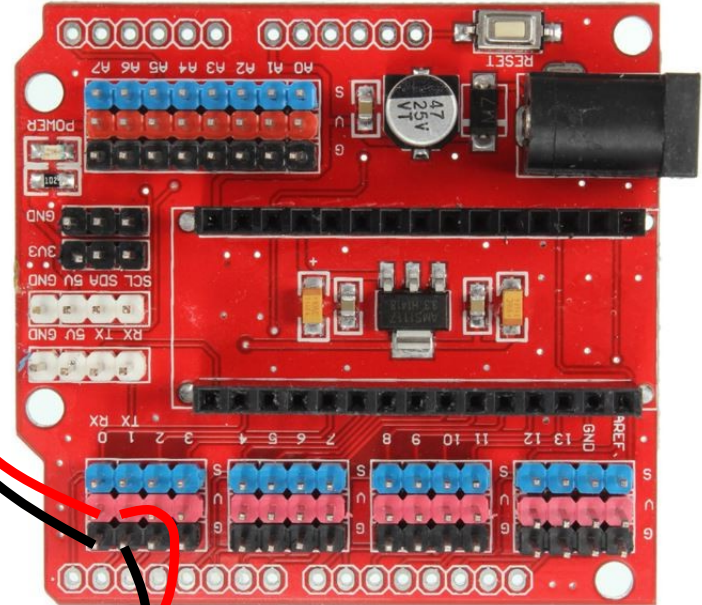
- Er zijn een stel korte draden (10 cm) en een stel langere (15 cm)
- Gebruik de kortst mogelijke draden
  - Geeft een minder rommelige opzet
  - Je komt anders mogelijk later lange draden te kort
- Negeer de kleuren van de draden.
- Je kunt in principe de schakeling opbouwen op verschillende plaatsen op het breadboard, echter ...
  - ... gebruik s.v.p. de aangegeven kolommen om te vermijden dat je later in de workshop ruimte te kort komt.

# Sluit de voedingsdraden aan

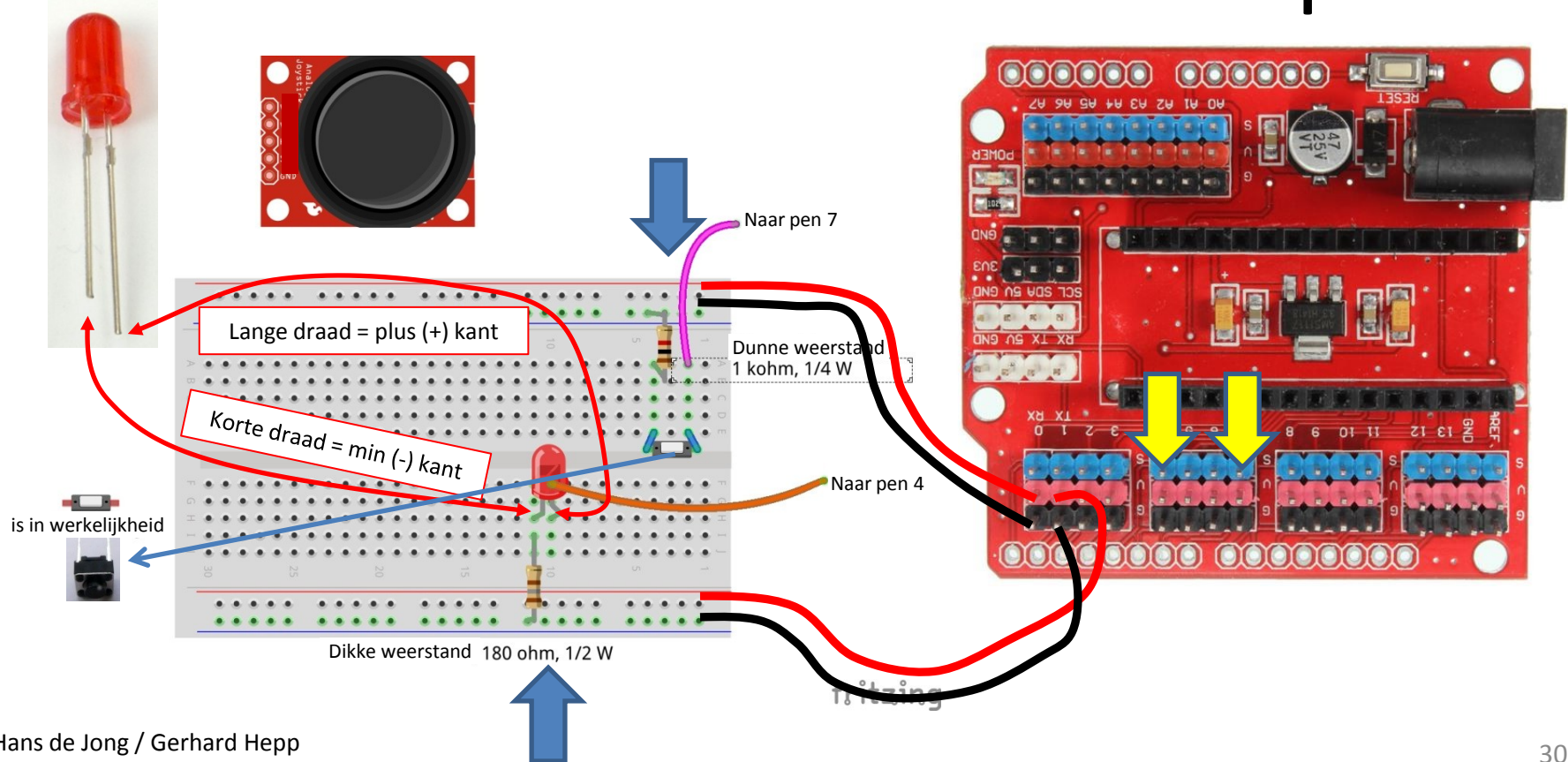
GND  
+5V  
VRx  
VRy  
SW



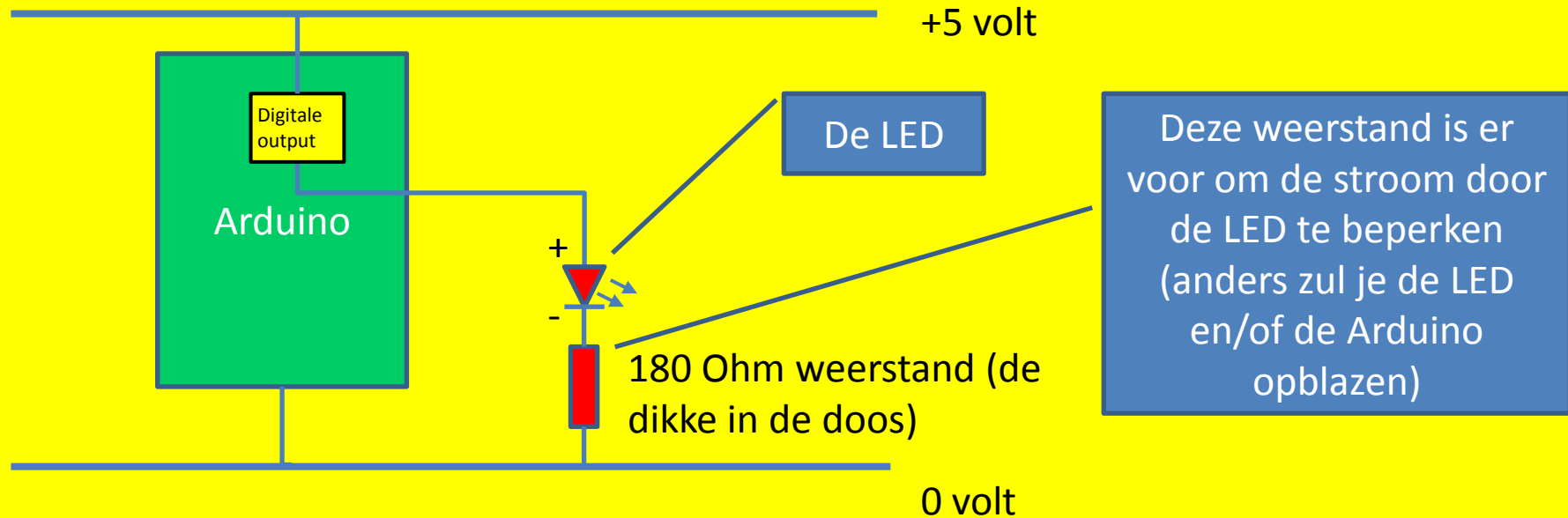
fritzing



# Plaats de rode LED en de knop



# Waarom zetten we een weerstand in serie met de LED?

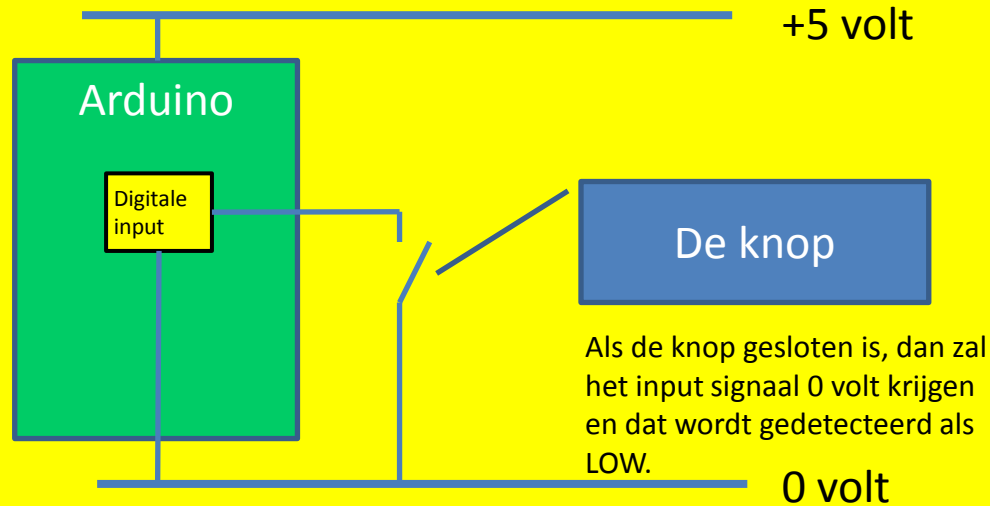


# Wat heeft een Arduino nodig op een Digitale Input pin?

- Dit moet een digitale input pin van Arduino krijgen:
  - 0 volt input (in feite, 0 tot 1.5 volt wordt beschouwd als een LOW input signaal)
  - 5 volt input (in feite, 2.5 volt tot 5 volt wordt beschouwd als een HIGH input signaal)
- Als het iets krijgt tussen 1.5 volt en 2.5 volt zal de interpretatie niet stabiel zijn (zou LOW of HIGH kunnen zijn).
- Als het helemaal geen signaal krijgt dan zal de interpretatie niet stabiel zijn (zou LOW of HIGH kunnen zijn).

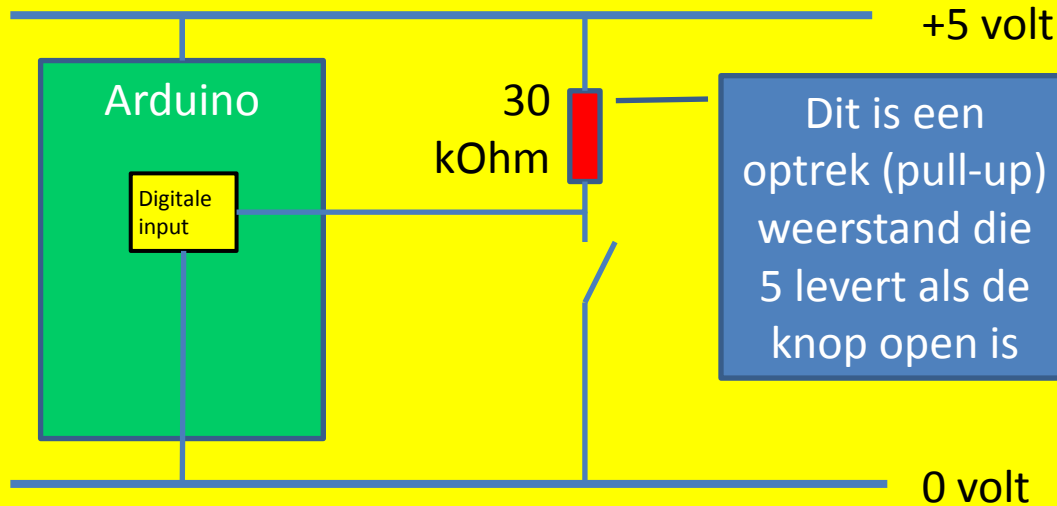


# Het maken van een LOW waarde



Maar wat detecteert de Arduino als de knop **open** is?

# Het maken van een HIGH waarde



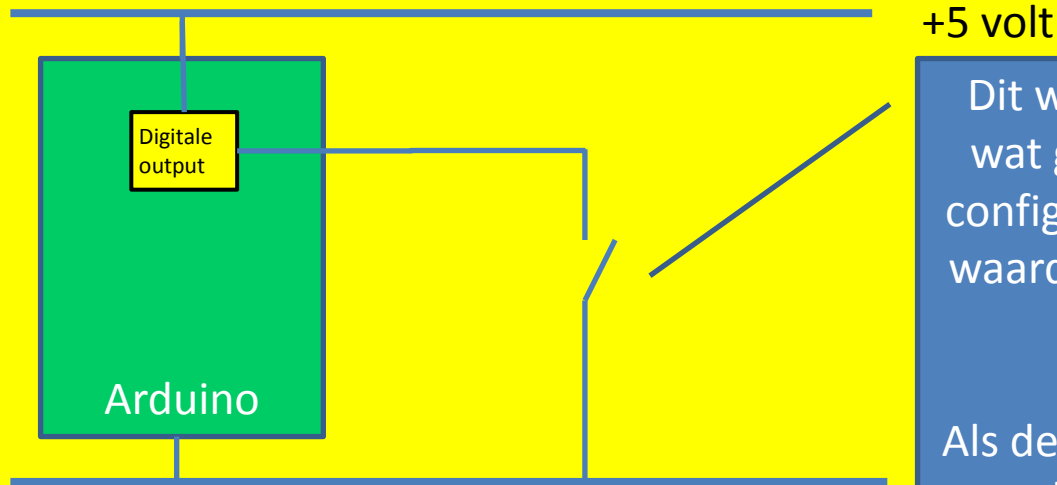
Als de knop ingedrukt is dan verbindt de knop de input van de Arduino naar 0 volt.

Als de knop open is zal de weerstand de input van de Arduino "optrekken" naar 5 volt. Er zal een kleine stroom door de weerstand en de digitale input op de chip lopen, maar is klein genoeg om te zorgen dat de spanning op de digitale input zeer dicht bij 5 volt is (= HIGH).

# Het maken van een HIGH waarde zonder externe weerstand



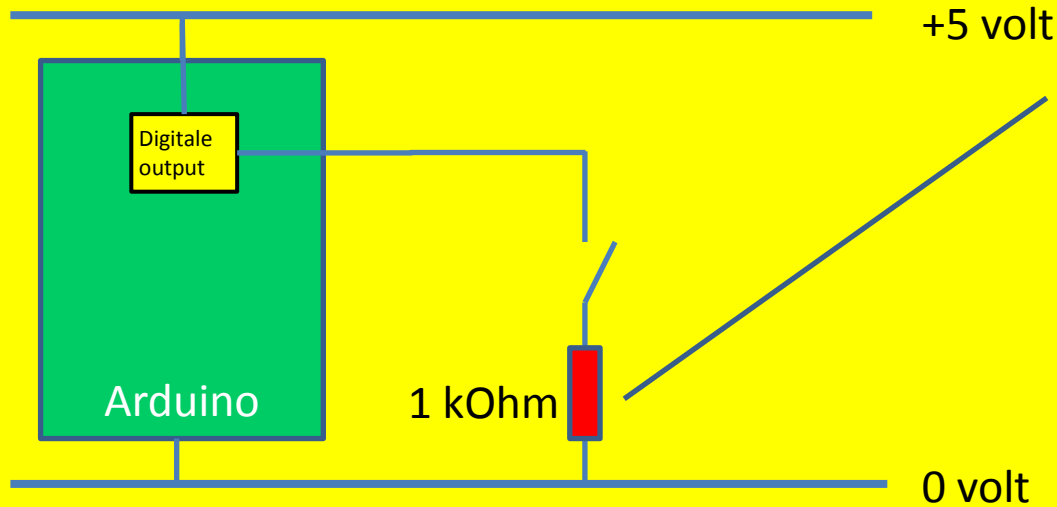
# Gevaar van fout configureren



Dit werkt prima voor input, maar wat gebeurt er als je de pin fout configureert als Digitale Output, de waarde op HIGH zet en op de knop drukt?

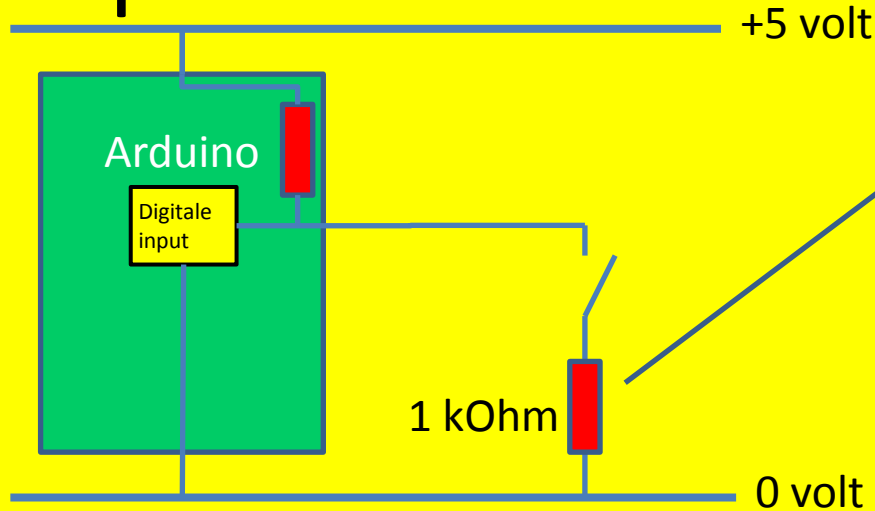
Als de Arduino de pin HIGH (5 volt) maakt en de knop verbindt naar LOW (0 volt) → kortsluiting en de Arduino zal opgeblazen worden.

# Een weerstand in serie met de knop voorkomt schade



Als de pen HIGH gemaakt wordt en de knop wordt ingedrukt dan zal er een kleine stroom lopen ter grootte van  $5 \text{ volt} / 1 \text{ kOhm} = 5 \text{ milliAmpere}$  die ver beneden het maximum van 20 mA.

# Als de poort correct is geconfigureerd als Digital In dan is een weerstand in serie prima



Zet daarom altijd een 1 kOhm weerstand in serie met de schakelaar.

Als de schakelaar gesloten is zal er een kleine stroom door de weerstanden lopen. Daarom zal de input  $1 \text{ kOhm} / (30 \text{ kOhm} + 1 \text{ kOhm}) * 5 \text{ volt} = 0,15 \text{ volt}$  zijn.

# Controleer en controleer nogmaals



- Controleer **beiden** dat de bedrading correct is.

# Breng de dingen samen

- Sluit de 9 volt connector aan
  - Hoewel zolang je alleen een LED gebruikt, de aansluiting via USB voldoende is
- Sluit de USB kabel aan op de Arduino
- Dubbelklik *scratchClient Tutorial.scl* op het bureaublad om scratchClient te starten met de config file die je net hebt bijgewerkt.
- Dan zal ook een browservenster starten waar je de variabelen kunt zien
  - Zie later in deze presentatie voor de uitleg
- In het terminalvenster zul je wat klachten zien dat scratchClient geen verbinding met Scratch heeft
  - Hetgeen logisch is omdat Scratch nog niet gestart is.





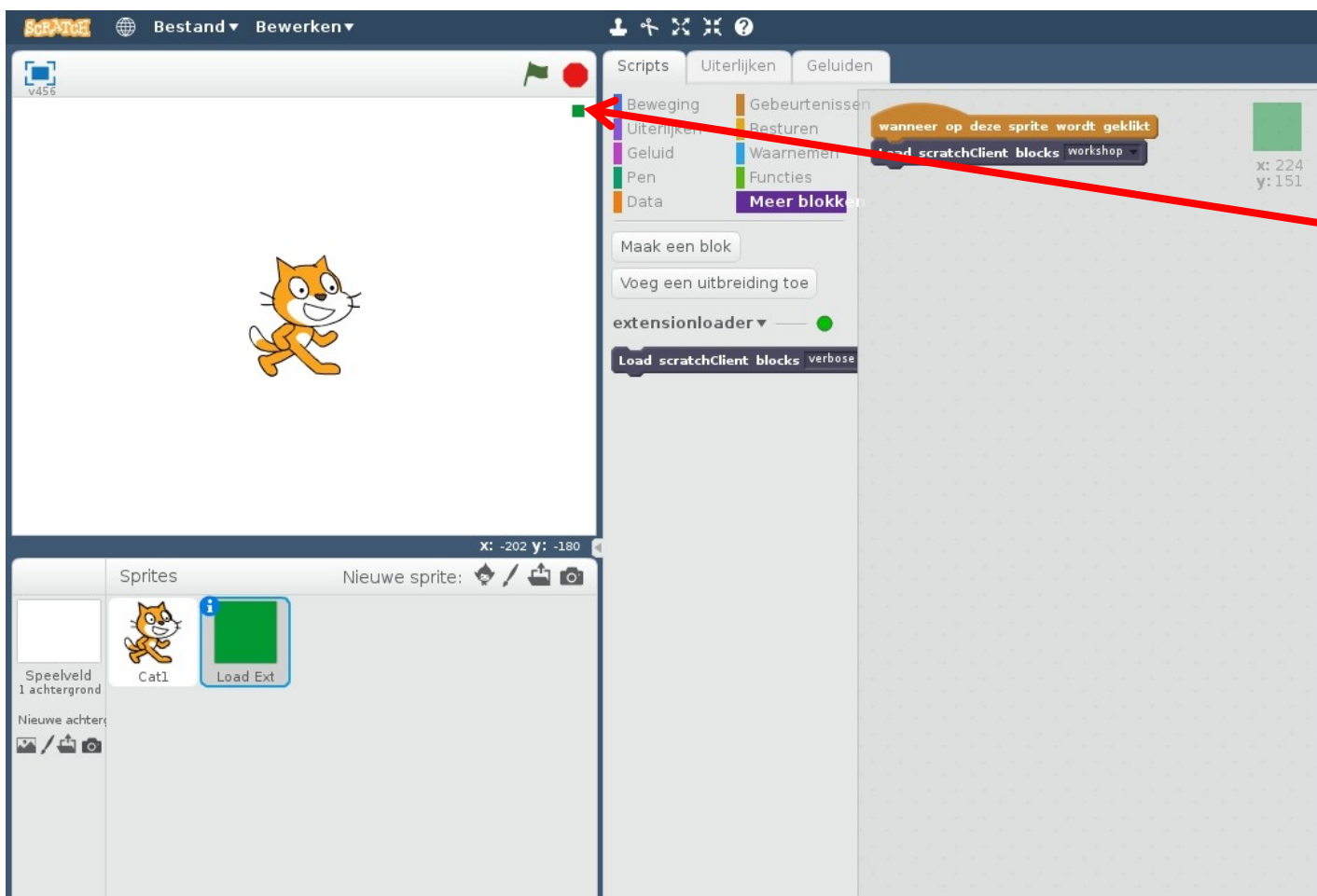
# Start Scratch 2

- Het simpelste is de file op het bureaublad te starten
- Dubbelklik het icoon

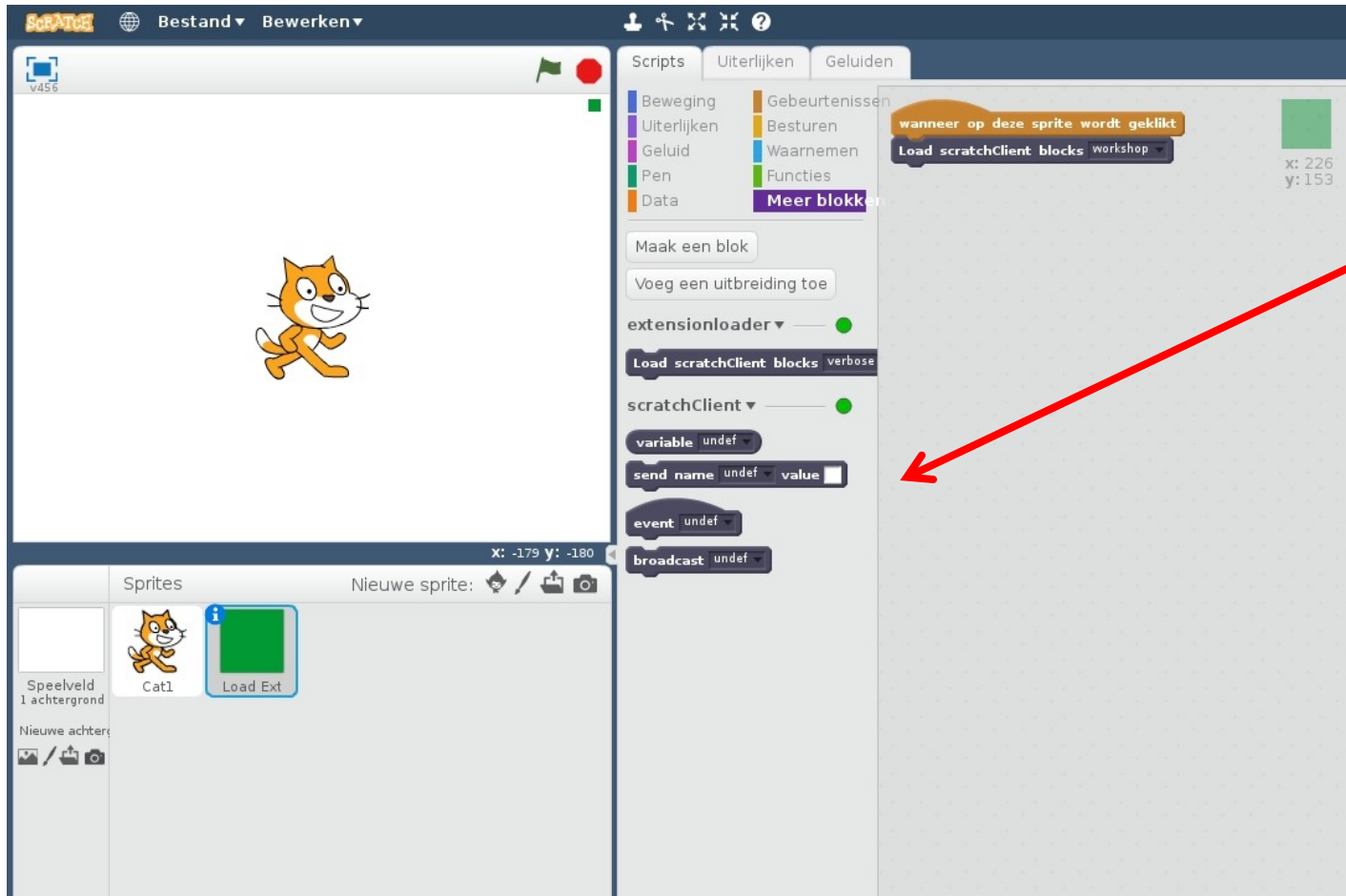


scratchClient-  
Tutorial.sb2

- Zie de bijlage hoe je zelf zo'n file kunt maken
  - Er zijn wat eigenaardigheden om rond heen te werken in het huidige release van Scratch 2



Klik op deze  
sprite om de  
extensie  
blokken  
zichtbaar te  
maken.



Het klikken op de groene sprite heeft er voor gezorgd dat deze extra blokken verschijnen.

# Maak het Scratch programma

- Maak dit programma in Scratch (zet het in de scherm of de Cat sprite)
- Probeer het uit (klik op de stapel blokken of klik op de groene vlag boven het animatie venster).
- Digitale in (Knop):
  - 0 = ingedrukt
  - 1 = niet ingedrukt
- Digitale uit (LED)
  - 0 = uit
  - 1 = aan
- Analyseer hoe het programma werkt.

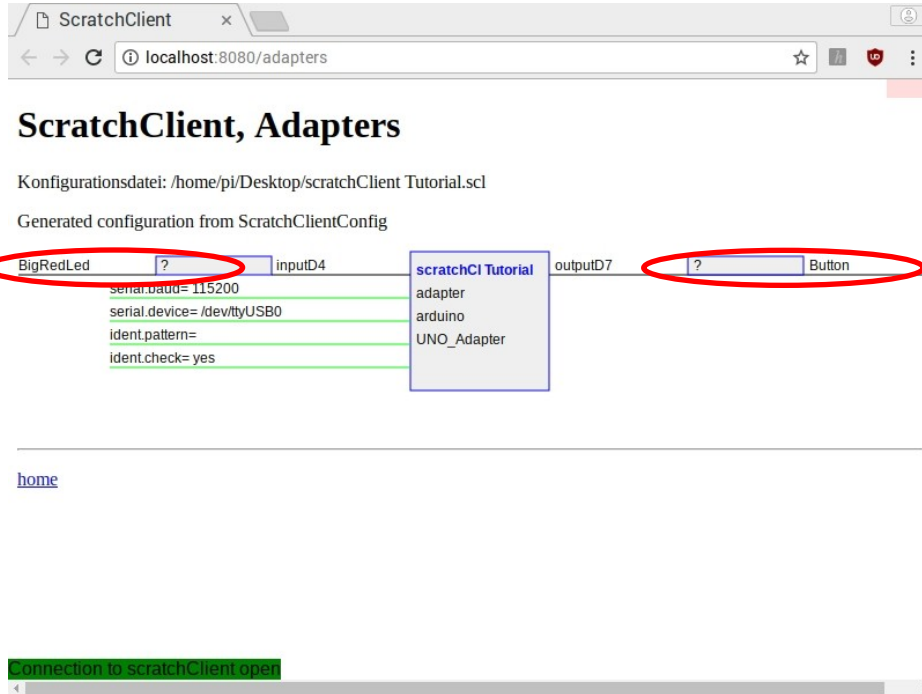


Gebruik deze pijlen om de correcte waarde te selecteren (het zwart op een grijze ondergrond maakt het niet zo goed zichtbaar)

# Werkt het? (zie de volgende pagina voor hulp)

- **Alleen** als de LED op de Arduino langzaam knippert dan is de config file gedownload en **alleen dan** werkt scratchClient.
  - Het kan 10 seconde duren nadat scratchClient en de scratchClient extensies in Scratch 2 geladen zijn voordat dit gebeurt.
  - Dus 10 seconde nadat je op de groene sprite hebt geklikt.

# Je kunt de waarden die uitgewisseld worden in de gaten houden



ScratchClient

localhost:8080/adapters

## ScratchClient, Adapters

Konfigurationsdatei: /home/pi/Desktop/scratchClient Tutorial.scl

Generated configuration from ScratchClientConfig

BigRedLed ? inputD4

serial.baud= 115200

serial.device= /dev/ttyUSB0

ident.pattern=

ident.check= yes

scratchCI Tutorial

adapter

arduino

UNO\_Adapter

outputD7 ? Button

[home](#)

connection to scratchClient open

- Tegelijk met scratchClient wordt de browser geopend met URL *localhost:8080/adapters*
- Let op de input & output richtingen:
  - De output van een adapter is een input voor Scratch.
  - De output van Scratch is een input voor de adapters.
  - Daarom lijken input en output verwisseld te zijn van wat in de config file staat.
  - Kijk daarom liever naar de namen van de variabelen.
- Je zult zien dat de waarden alleen worden weergegeven nadat ze gewijzigd zijn (anders staat er een vraagteken (?) ).
- Niet voor nu, maar je kunt op een veld klikken en de waarde intikken die dan in de juiste richting worden gezonden.

# Wijzig het programma

- Maak een wijziging in het programma zodat de LED aangaat als de knop ingedrukt wordt.
- Er zijn (minstens) twee manieren
- Je hebt daar deze blokken voor nodig



Blokken voor methode 1



Blokken voor methode 2

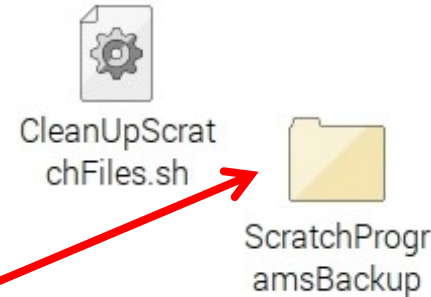
# Wat als het niet werkt?

- Controleer dat je niet meer Scratch instanties open hebt
  - scratchClient kan alleen werken met een Scratch tegelijk open (ongeacht of dit Scratch 1.4 of Scratch 2 is).
- Controleer dat de blauwe LED op de Arduino Nano *langzaam* knippert.
- Soms, speciaal na reboot, als alles verder goed lijkt, kan het helpen om de Arduino Nano af te koppelen en weer aan te sluiten.
- Kijk naar waarden van de variabelen in de browser, zie eerder.



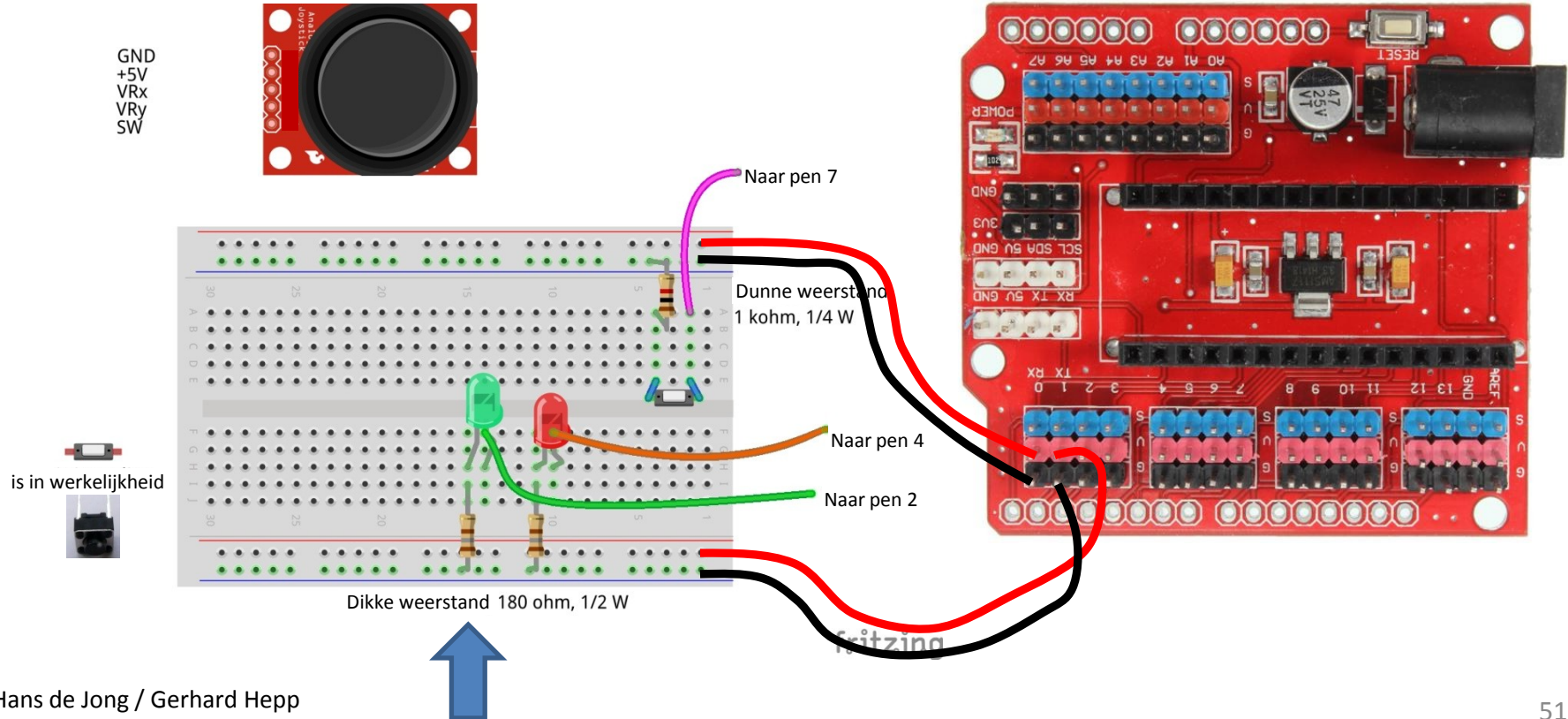
# Herinnering ...

- Sla je werk regelmatig op. Anders gaat het verloren als de stroom uitvalt.
  - En stroomuitval kan gemakkelijk gebeuren omdat je kabels erin en eruit doet en dat kan ook de voeding van de Raspberry Pi beïnvloeden.
- Hier zijn wat merkwaardigheden van Scratch 2 waar we omheen moeten werken
  - Gebruik geen spaties in bestandsnamen (als je dat doet, dan kan Scratch 2 de file niet vinden als je erop dubbelklikt).
  - Scratch 2 vergeet de map waar de file uit geopend was (Bureaublad). Sla het toch op in wat voorgesteld wordt: */home/pi*
  - Scratch 2 slaat het op met een *.sbx* extensie ongeacht wat je opgeeft.
- Daarom, eenmalig na opstarten van de Raspberry Pi
  - Op het bureaublad, dubbelklik *CleanUpScratchFiles.sh* 
  - Je kunt het venster klein maken, maar laat het altijd lopen
  - Dit zorgt dat er permanent wordt gekeken naar de map */home/pi*, en
    - Alle bestanden met *.sbx* and *.sb2* extensie worden verplaatst naar het bureaublad als *.sb2* bestanden
    - Alle spaties uit de bestandsnaam worden verwijderd
    - Als er al zo'n bestand op het bureaublad aanwezig is wordt het oude bestand verplaatst naar de map *ScratchProgramsBackup* op het bureaublad 
- Als je het opgeslagen bestand opnieuw opent, klik dan op de groene sprite om te zorgen dat de extensie blokken weer worden geladen.



# Deel 6: Toevoegen van de grote groene LED

# Toevoegen van de groene LED



# Controleer en sluit weer aan

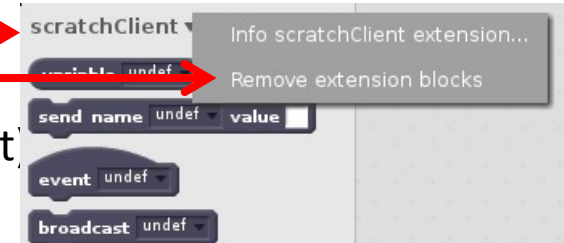
- Controleer de bedrading
- Sluit de 9V adapter aan
- Sluit de USB kabel aan

# Werk de config file bij en herstart scratchClient


- Gebruik de config tool (die nog steeds open zou moeten zijn)
- Definieer een output (direction: out, function: output) op pen 2 en noem die *BigGreenLED*
- Sla de config file op (en laat de config tool open)
- Dubbelklik *scratchClient Tutorial.scl* op het bureaublad
- Dit stopt de vorige instantie van scratchClient en herstart het met de bijgewerkte configuratie.
- Verwijder het extensieblok.
  - Klik rechts hier
  - Kies *Remove extension blocks*
- Voeg het extensieblok weer toe (klik op het groene vierkant)
  - Je zult in de *More Blocks* sectie zien dat je nu ook kunt kiezen om waarden te zenden naar BigGreenLED



scratchClient-  
Tutorial.scl



# Werk Scratch bij

- Voeg code toe dat dit doet:
  - Druk op de knop: LED gaat aan
  - Druk nogmaals: LED gaat uit
- Je hebt dit nodig. 



# Dit is het eind van de beginners workshop

- Je hebt *digitale output* (LED) en *digitale input* (een knop) werkend
- Je weet nu hoe te werken met scratchClient en Scratch 2
- Je weet hoe scratchClient te configureren
- Als je zin en tijd over hebt
  - Kijk naar sommige van de gele pagina's als je ze eerder overgeslagen hebt
  - Ga verder met een van de volgende niveau's.
    - Deze zitten in andere bestanden.
- Anders:
  - Kopieer het materiaal naar je USB stick en ruim op. Zie verder hieronder.

# Overzicht van de volgende niveau's

- Gevorderden niveau
  - Analooog Input: Potentiometer
  - Pulsbreedte modulatie (Pulse Width Modulation, PWM, in afwezigheid van analooog output)
    - Dimmen van een LED
    - Besturen van een servo
    - Besturen van een zoemer
- Geavanceerd niveau
- Expert niveau



# Deel 10: Neem je werk mee naar huis

# Wil je je werk mee naar huis nemen?

- Als je je eigen USB stick hebt meegebracht, stop die dan in de Raspberry Pi en kopieer het *scratchClient-Tutorial.scl* bestand op het bureaublad, plus alle .sb2 bestanden die je op het bureaublad hebt gemaakt.
- De rest van het materiaal kun je downloaden van [www.github.com](https://www.github.com), zoek naar *scratchClient-Tutorial*
- Neem het blaadje mee zodat je weet waar je het materiaal op github kunt vinden.

# Deel 11: Samenvatting en wat te onthouden

# Onthouden uit beginners workshop

- Met scratchClient kun je definiëren:
  - De functie van elke pen
  - Een symbolische naam voor elke pen
- scratchClient config is het programma om de configuratie te doen
- Herstart scratchClient nadat je de configuratie hebt veranderd
- Zet een weerstand in serie met de LED's
- Zet een weerstand in serie met de schakelaars
- Configureer een optrekweerstand (pull up resistor) als het signaal varieert tussen 0 volt en open in plaats van tussen 0 volt en 3 tot 5 volt.
- In Scratch 2, gebruik extensieblokken voor scratchClient om blokken te krijgen waarmee je met scratchClient kunt werken
- Je kunt de waarden van alle pennen in de gaten houden in de browser
- **scratchClient kan nog veel meer doen ...**
- Er zijn wat merkwaardigheden, daarom
  - Sla de Scratch 2 bestanden op in de thuismap (zoals Scratch 2 voorstelt)
  - Laat *CleanUpScratchFiles.sh* lopen. Die staat op het bureaublad. Eenmalig na opstart van de RPi.
  - Werk vanaf het bureaublad
  - Klik op het groene vierkantje elke keer nadat je een project met scratchClient heropent.
- Een pen op Arduino kan deze functies hebben:
  - Digitaal In
  - Digitaal Uit
  - Analooq In \*
  - *Geen Analooq Uit*
  - Pulsbreedte modulatie (Pulse Width Modulation, PWM) als alternatief voor het ontbreken van Analooq Uit \*
    - Voor het sturen van de intensiteit van een LED
    - Voor het besturen van een servo
    - Voor het besturen van een zoemer
  - Er zijn een paar meer, zie het geavanceerde niveau
  - Je kunt optrekweerstanden (pull up resistors) configureren op de Digitale In

\* Zie het gevorderden niveau

# Deel 12: Afbreken en opruimen

# Als je scratchClient dag hier eindigt...

- Koppel de USB kabel en de 9V adapter af
- Haal alle componenten en draden uit het **breadboard**
- Trek alle draden uit het **Arduino bord**.
- **Laat de draden aan de 3-kleuren LED zitten** (die heb je niet gebruikt)
- **Laat de draden aan de zoemer zitten**
- Als iets kapot is, dan s.v.p.
  - Gooi het weg of lever het in (als het onduidelijk is)
  - Doe een briefje in de doos met daarop wat er mist
  - Doe nooit iets dat kapot of mogelijk kapot is terug in de doos
- Sluit de Raspberry Pi af
- Laat ons weten wat je van de workshop vond, nu mondeling of later per email
  - [hans.piam@hanselma.nl](mailto:hans.piam@hanselma.nl)

# Bijlage A

## Extensie blokken in Scratch 2

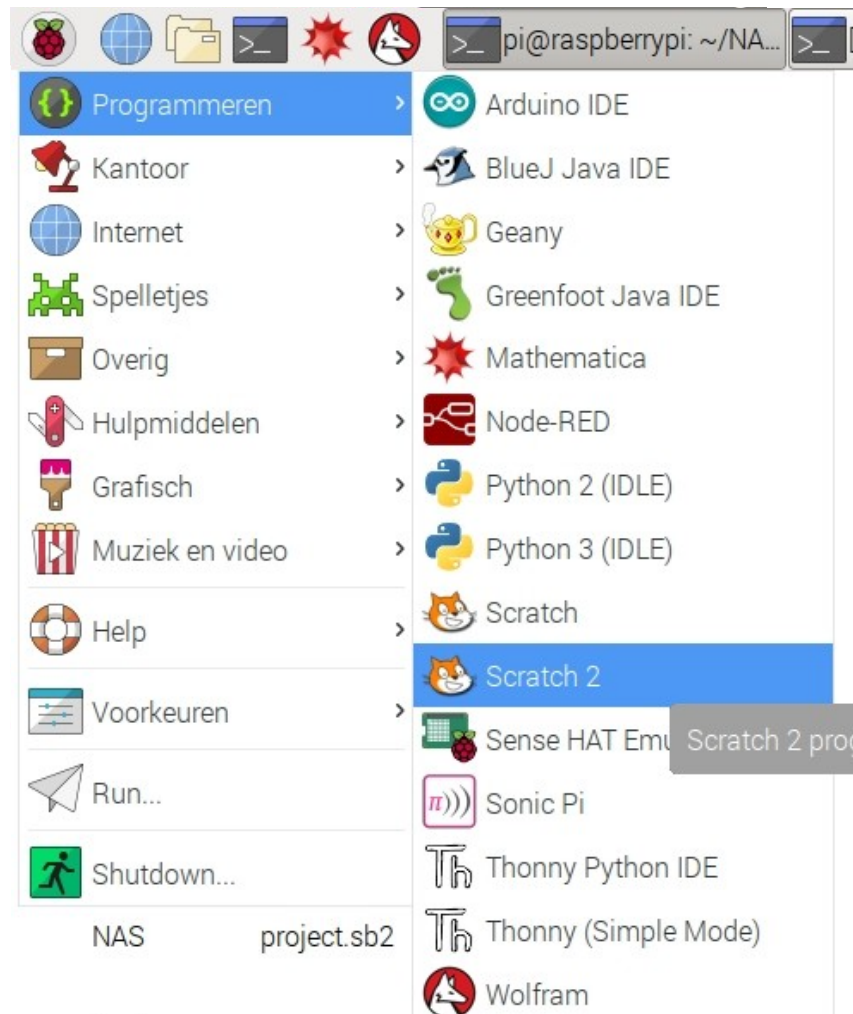
# Hoe moet je extensieblokken toevoegen?

- Wegens een aantal merkwaardigheden is het laden van extensieblokken enigszins omslachtig.
- Je kunt de eerder uitgelegde manier gebruiken (klik op de groene stip sprite)
- Deze bijlage legt uit hoe dit uit Scratch te doen.
- Stap 1: Zorg dat scratchClient is loopt.



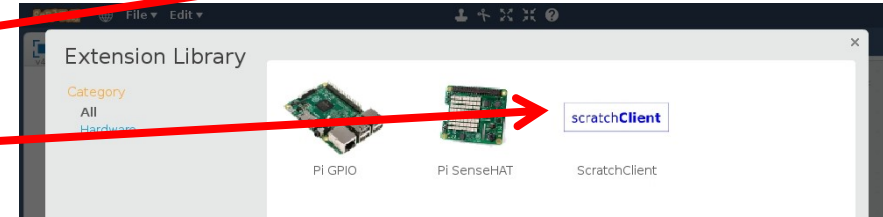
# Stap 2:

## Start Scratch 2

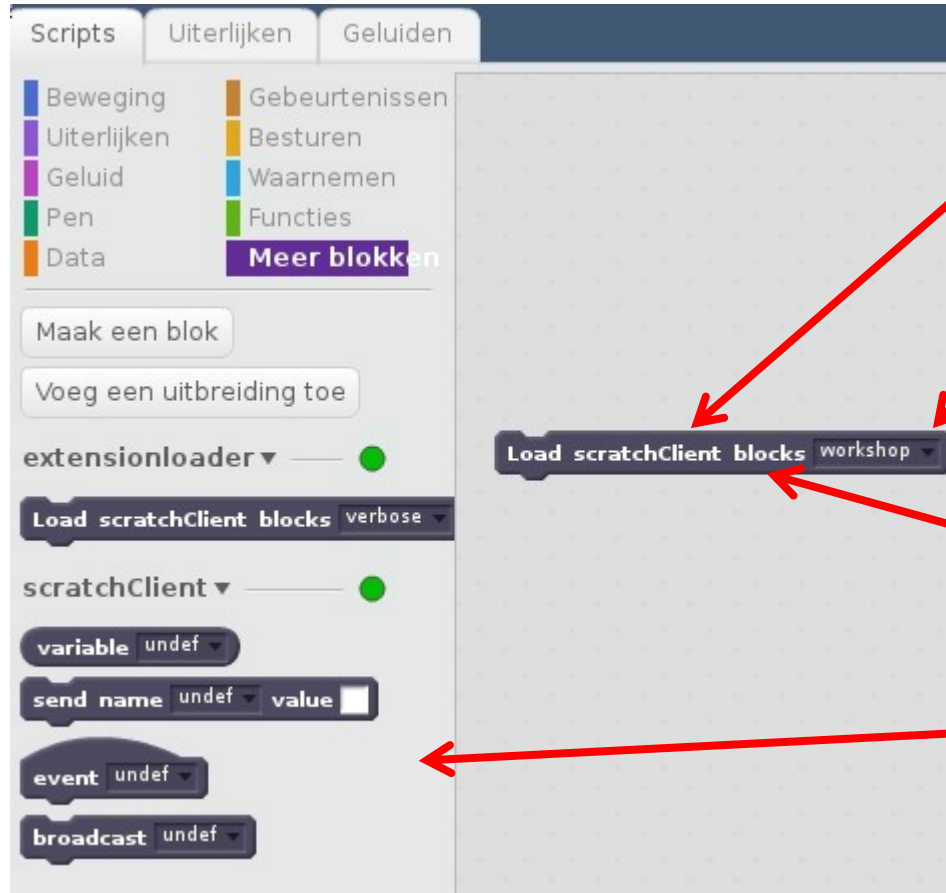


# Stap 3: Laad scratchClient blokken (1 van 2)

- Klik op *More Blocks*
- Klik op *Add an Extension*
- Kies *scratchClient*
- Nu wordt er een laadprogramma voor de extra blokken van scratchClient get opgenomen.
- Maar nu moeten we nog de echte blokken laden ...



# Stap 4-7: Laad scratchClient blocks (2 van 2)



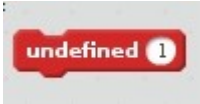
Stap 4: Sleep dit blok naar het Scripts gebied

Stap 5: Kies hier *workshop*

Stap 6: Klik op het blok zodat de extra blokken verschijnen.

Stap 7: Zie de blokken verschijnen.

# De blokken worden niet opgeslagen

- Als je het bestand opnieuw opent nadat die opgeslagen was, dan
  - Is de scratchClient extension loader nog steeds aanwezig
  - Zijn de echte blokken verdwenen
  - En zie je veel blokken zo: 
- Daarom moet je na heropenen stappen 4 – 7 opnieuw doen.
- Voor een andere methode, zie de volgende pagina.

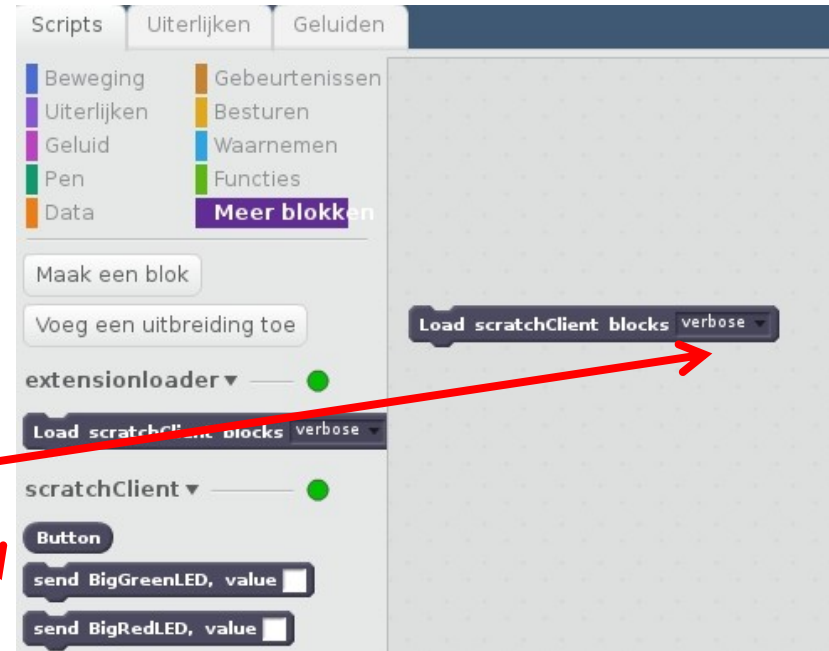


# Alternatief voor for stappen 4 – 7

- In plaats van stappen 4 – 7 elke keer handmatig te doen, kun je:
- Een sprite opnemen  
*Load Name-Value Extension Blocks.sprite2* uit de map  
*~/scratchClient-Tutorials/scratchClientExtension/Sprites*
  - *Bedenk dat bij het laden van een sprite in Scratch 2 op RPi de muisknoppen soms dood zijn en dat je zult moeten navigeren met de pijltjes (selecteren met return / enter). Esc haalt je uit die toestand.*
- Deze sprite heeft een klein groen vierkantje dat je kunt klikken om de extra blokken te laden elke keer dat je de file opnieuw opent.

# Een andere representatie van blokken

- scratchClient blokken kunnen ook weergegeven worden met een blok per input of output waarde.
- Je krijgt dan b.v.
  - Een apart blok voor *BigRedLED*
  - Een apart blok voor *BigGreenLED*
  - Een apart blok voor elke input variabele (hier alleen *Button*)
- Je kunt deze weergave krijgen door *verbose* te selecteren in het load blok



# Makkelijke manier om de *verbose* versie van de blokken te krijgen

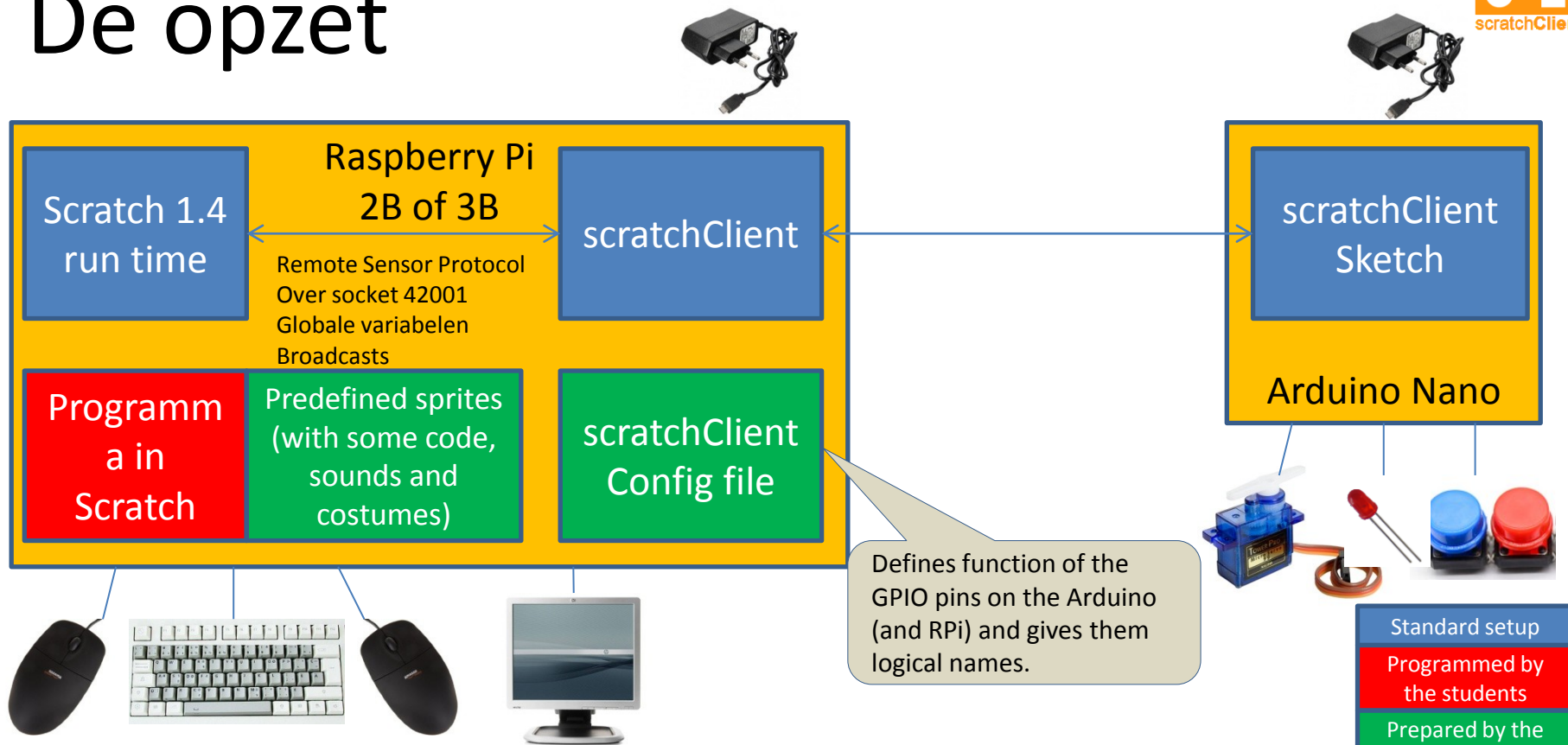
- Neem de sprite  
*Load Verbose Extension Blocks.sprite2* uit de map  
*~/scratchClient-Tutorials/scratchClientExtension/Sprites* op
  - *Bedenk weer dat de muisknoppen mogelijk dood zijn bij het laden, zie eerder.*
- Die sprite heeft een kleine groene cirkel die je kunt klikken om de extra blokken te krijgen elke keer dat je een bestand opnieuw opent.
  - Vergelijk dat met het laden van de naam-waarde paren waarvoor je de andere sprite met het kleine groene vierkantje kunt gebruiken.

# Bijlage B

## Als je Scratch 1.4 gebruikt




# De opzet

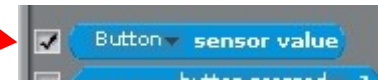
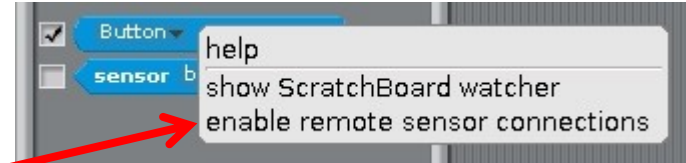


# Definiëren van de config file

- Niets nieuws onder de zon, je kunt precies dezelfde config file gebruiken voor Scratch 1.4 als voor Scratch 2.

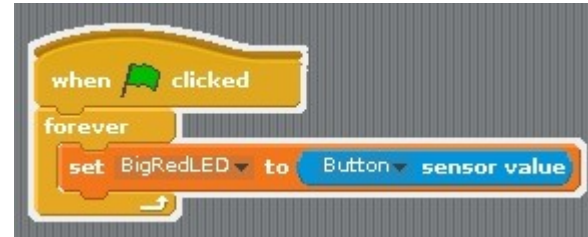
# Maak het Scratch programma

- Start Scratch  → Programming → Scratch
- Enable remote connections (right click on *sensor value*)
- Create the variable *BigRedLED*, available to all sprites
- Make the variable visible (tick the box in front)
- Make the *Button* sensor visible
- Save the file on the desktop.



# Programma in Scratch 1.4

- Maak dit programma in Scratch dat de BigRedLED laat oplichten als de knop losgelaten wordt.
- Test of het werkt.



# Verder ...

- Je kunt nu verder kijken naar het Scratch 2 materiaal eerder in deze presentatie.
- Probeer uit om de BigGreenLED toe te voegen en dezelfde blokken om daarmee te werken.

# Bijlage C

## Basis elektronica

# Nog toe te voegen

# Appendix D

## Meer informatie



# Meer informatie

- Al het workshop materiaal
  - [www.github.com](https://www.github.com) and zoek naar *scratchClient Tutorials*
- scratchClient
  - [http://heppg.de/ikg/wordpress/?page\\_id=6](http://heppg.de/ikg/wordpress/?page_id=6)
- Scratch
  - <https://scratch.mit.edu/>
- Scratch on Raspberry Pi
  - <https://www.raspberrypi.org/forums/viewforum.php?f=77>
- Raspberry Pi
  - <https://www.raspberrypi.org/>
- Arduino
  - <https://www.arduino.cc/>

# Eind van de **beginners workshop**